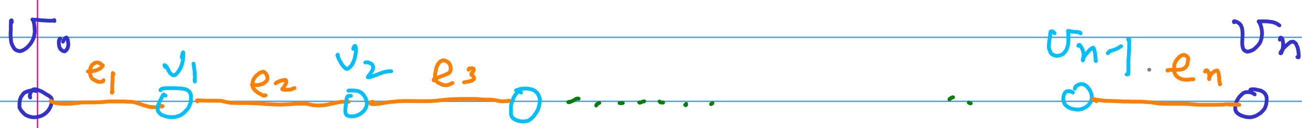
Graph (H1)

20150612

Path

$v_0, \cdots, v_n$: Vertices (정점)



node ○ : $n+1$ 개 교대로 alternating
edge — : n 개

결합 (association)

e_i 는 v_i 에 결합되어있다
e_i 는 v_{i-1} 에 결합되어있다

Connected Graph

a Vertex $\rightsquigarrow$ another vertex

어떤 a path

Graph G for any 2 vertices v, w

if $v \longrightarrow w$ a path exists

then Graph G : "connected"

Subgraph of G

$\boxed{G = (V, E)}$

꼭지점 집합   edge 집합

$\boxed{G' = (V', E')}$

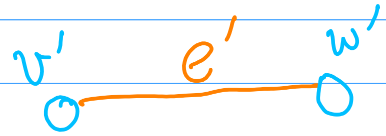
① $V \supseteq V'$, $E \supseteq E'$

② for every edge $e' \in E'$

Associated vertices must be included

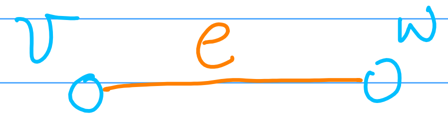
$v' \in V'$

$w' \in V'$

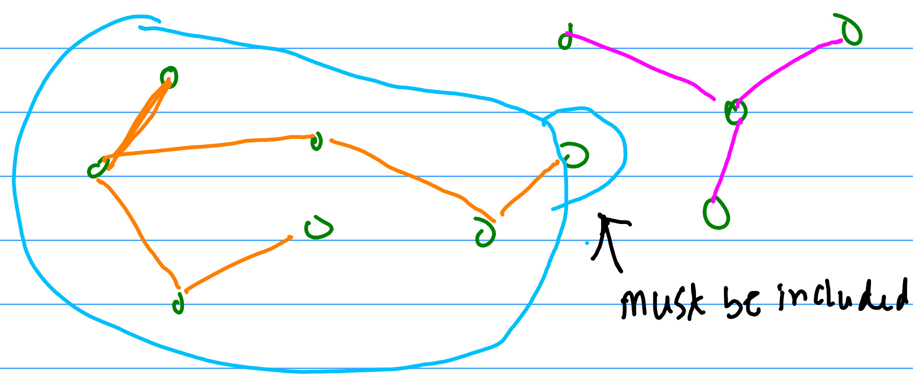


Subgraph G' 만들기

if edge e is included in a subgraph G'



vertices v, w that are associated with e must be included in G'



꼭지점 집합 edge 집합

원래 graph $G = (V, E)$

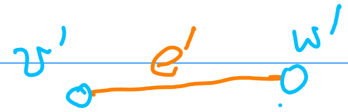
부분 graph $G' = (V', E')$

의 conditions

① $V' \subseteq V$

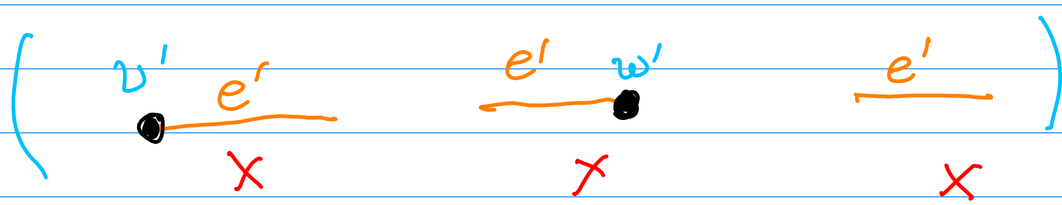
$E' \subseteq E$

② E'에 속한 모든 edge e'에 대해서



$v' \in V'$

$w' \in V'$



$G=(V,E)$

$V=\{v_1, v_2\}$ $\quad V' \rightarrow \phi, \{v_1\}, \{v_2\}, \{v_1,v_2\}$

$E=\{e_1\}$ $\quad E' \rightarrow \phi, \{e_1\}$

$G_1=(\{v_1\}, \phi)$

$G_2=(\{v_2\}, \phi)$

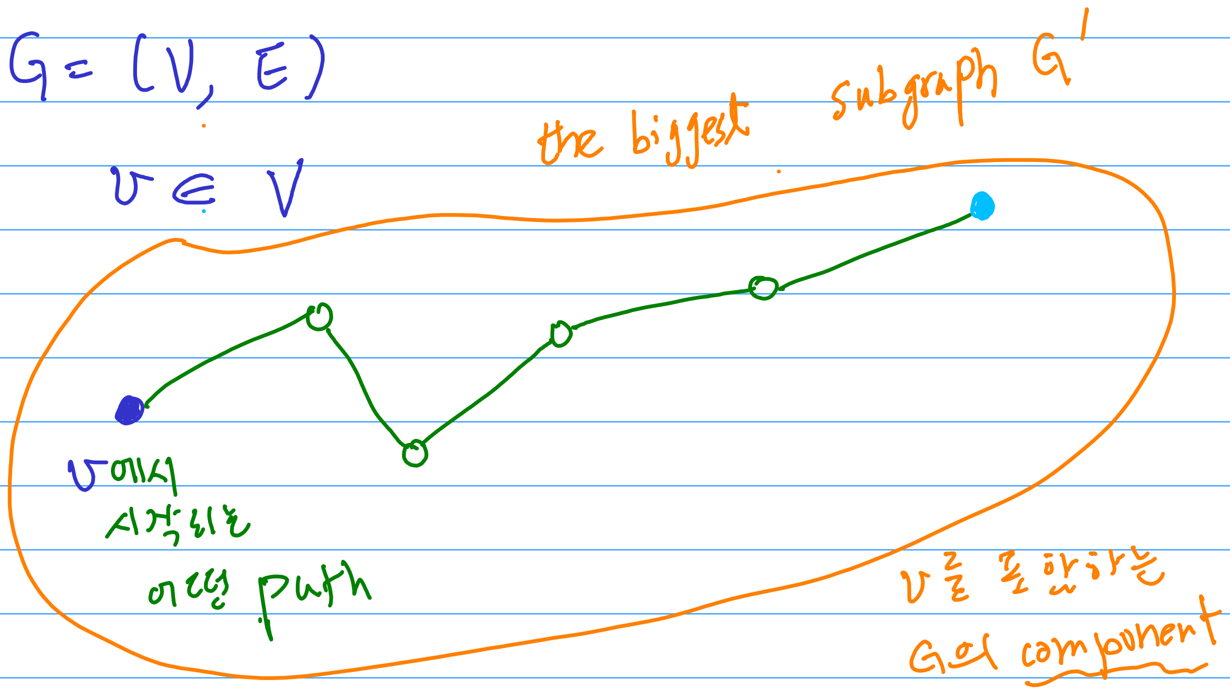
$G_3=(\{v_1,v_2\}, \phi)$

$G_4=(\{v_1, v_2\}, e_1)$

정의 8.2.11 G의 component

$G = (V, E)$

$v \in V$



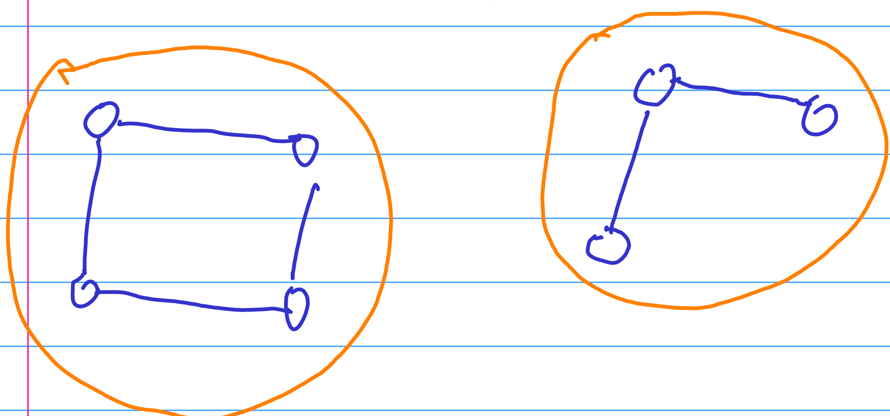
v에서 시작되는 어떤 경로에 포함된
G안의 모든 edge와 vertex로 구성된
G의 부분 graph G' → G의 component

a subgraph in which

any two vertices are connected to each other by paths

and which is connected to no additional vertices
in the subgraph

maximally connected

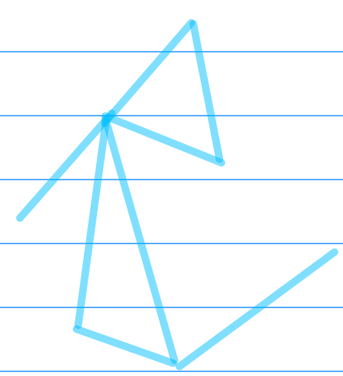


정의 8.2.11 Component 란?

graph G 에서 아무런 vertex v

v 에서 시작되는 모든 경로에 포함되는
모든 edge 와 vertex 들로
구성된 부분 graph G' 이다

Let G be a graph and let v be a vertex in G. The subgraph of G consisting of all edges and vertices in G that are contained in some path beginning at v is called a component of G containing v.



1개 component

v에서 시작되는 어떤 경로 상에 있는 모든 edge와 vertex로 구성된 G의 subgraph

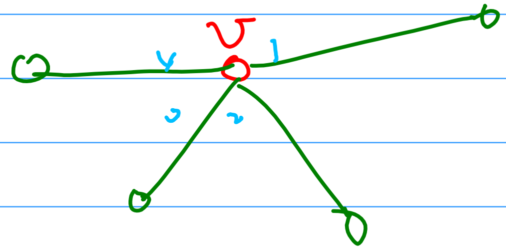
G의 subgraph 인데 이 subgraph 의 edge와 vertex들이
항상 v에서 시작되는 G의 어떤 경로 상에 있을 때

Connected graph $\Longleftrightarrow$ 1개 component

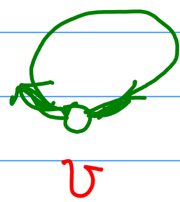
필요충분조건

degree of a vertex v $\delta(v)$

v에 결합된 vertices의 갯수



$\boxed{\delta(v) = 4}$

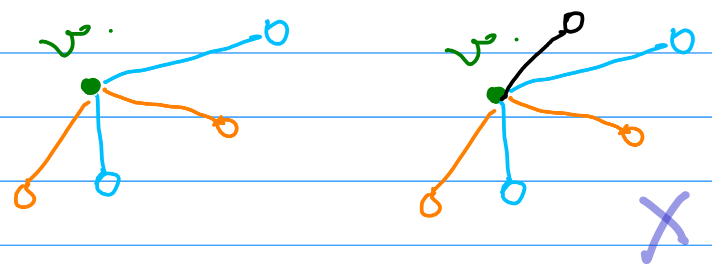


$\delta(v) = 2$

Theorem 8.2.17

Graph G has a <u>Eulerian cycle</u>

$\Longrightarrow$ Graph G $\begin{cases} \text{connected} \\ \text{모든 vertex } v \quad \underset{\text{degree of a vertex } v}{\delta(v)} : \text{even} \end{cases}$



Theorem 8.2.18

Graph G $\begin{cases} \text{connected} \\ \text{모든 vertex } v \quad \delta(v) : \text{even} \end{cases}$

$\Longrightarrow$ Graph G has an Eulerian cycle

정리 8.2.17

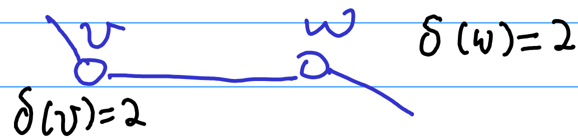
G has Euler cycle

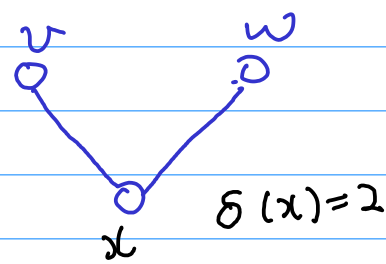
$\Rightarrow$ G : connected graph (1 component)

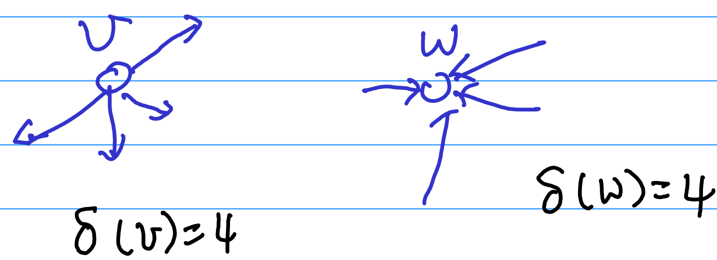
$\delta(v_i) =$ 짝수. for any vertex v_i

G의 any two vertices : v, w

Euler cycle







8.2.18

G : connected graph (1 component)

$\delta(v_i)$ = 짝수. for any vertex v_i

$\Rightarrow$ G는 Euler cycle을 가진다.

\# of edge = n

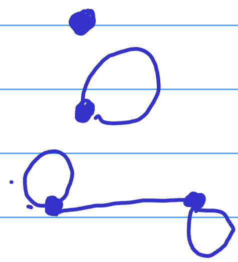
$n=0$ no edge $\rightarrow$ one vertex

G has n edges

$k < n$
짝수차수 정점
connected graph
). 가 Euler cycle을 가진다고 가정

n : # of edges

$n=0$



짝수 차수

$n=1$

$n=2$

...

Graph G가 n 개의 edge를 가진다 $\quad n > k$

> k 개의 edge를 가지고
> 모든 vertex가 even degree 이고
> connected graph $\Longrightarrow$
> Euler cycle을 가진다

assumption

귀납법

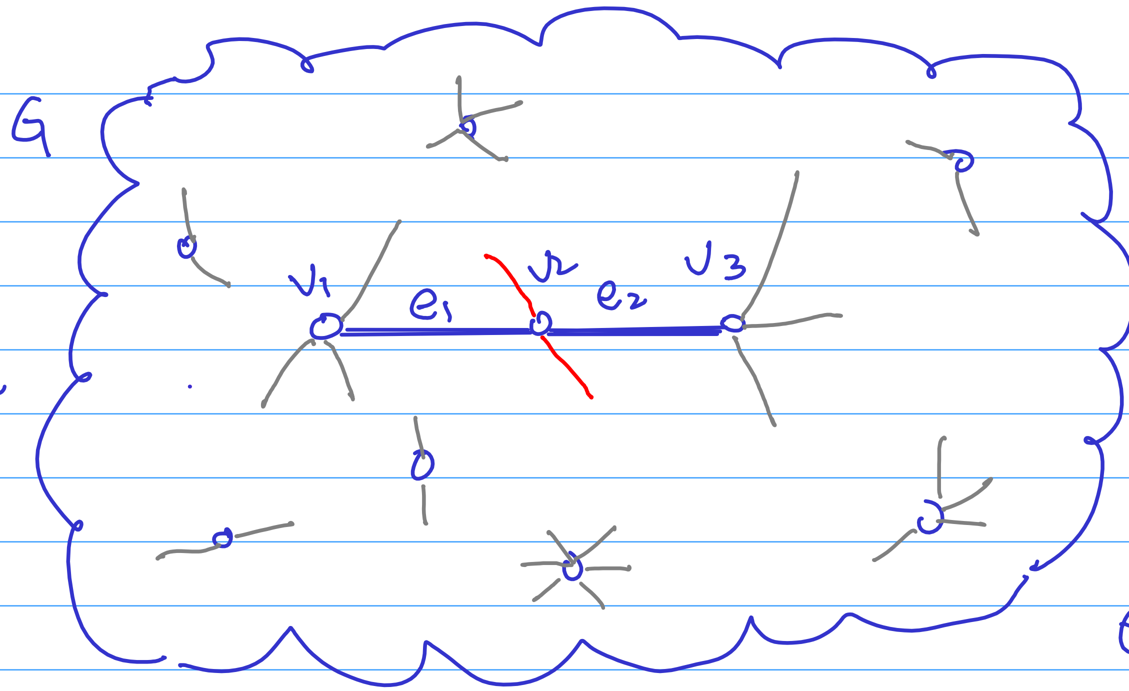
n 보다 작은 경우의

k 개 edge graph에서
- even degree
- connected graph

$\Longrightarrow$ Euler cycle 있다

n 개 edge graph에서
- even degree
- connected graph

$\Longrightarrow$ Euler cycle 있다



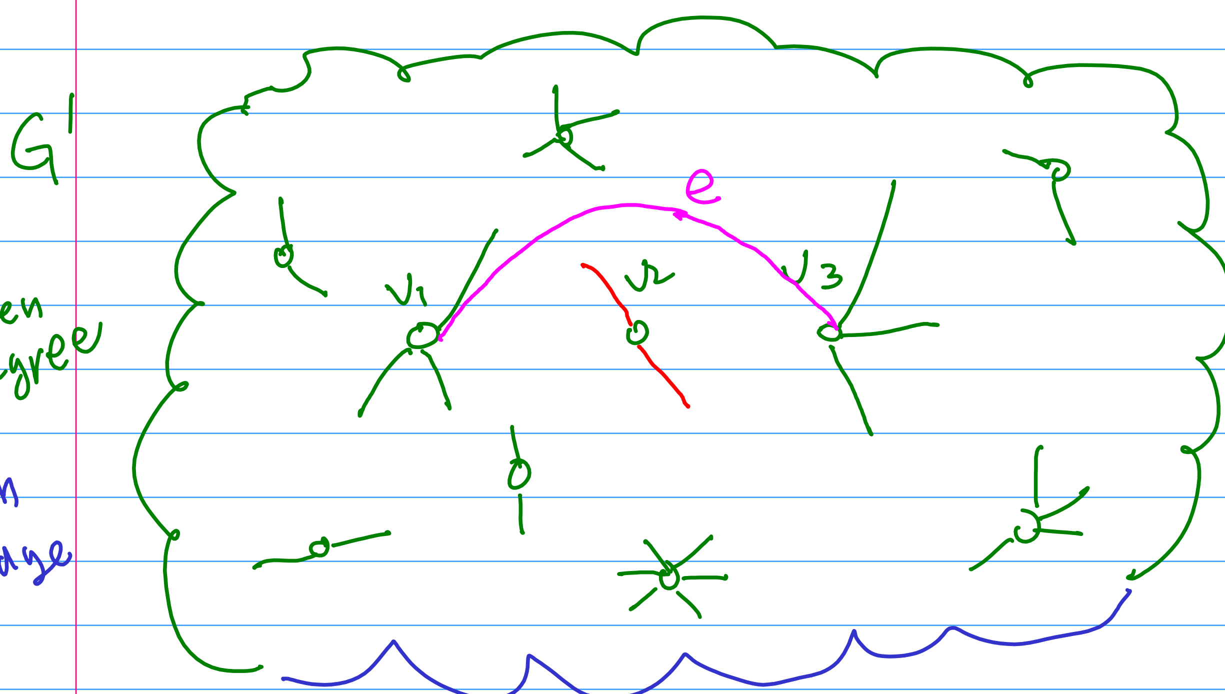
even degree

n $m+1$ edge

e_1, e_2

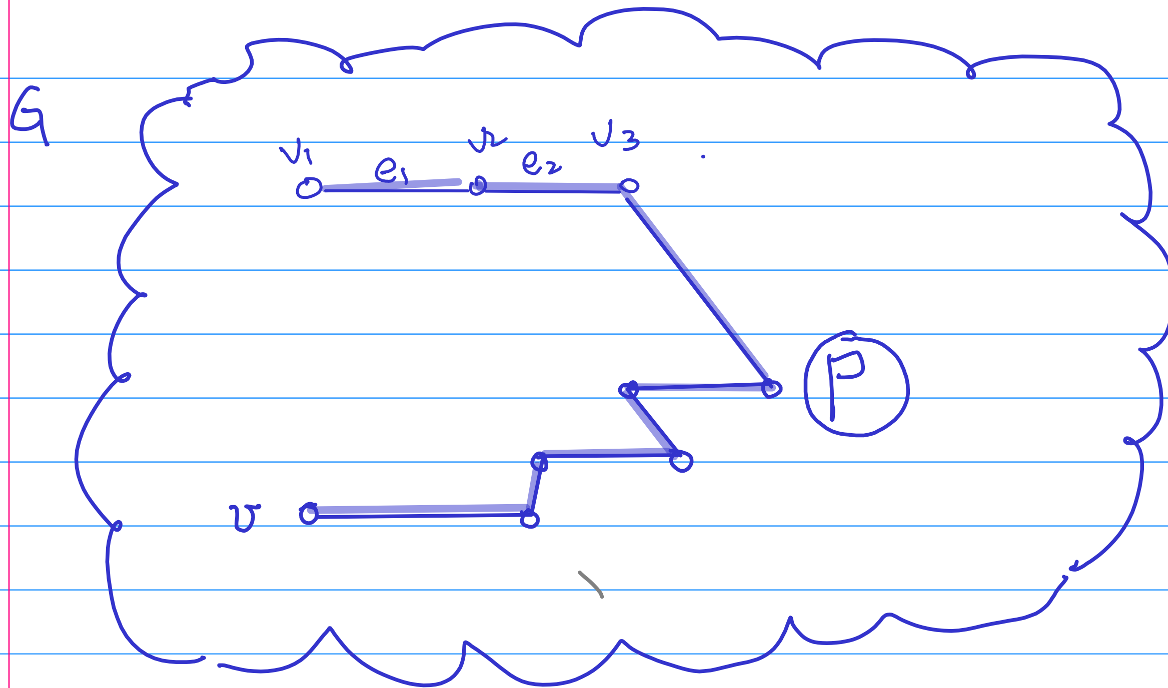
replace

e

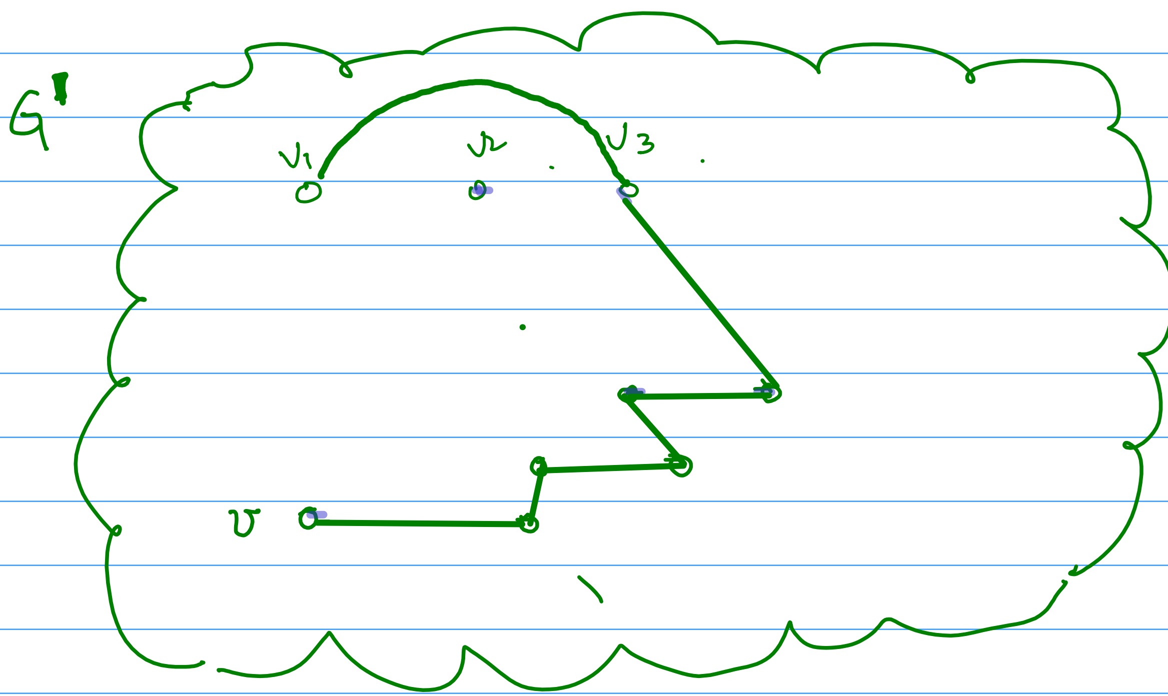


even degree

n 1 m edge



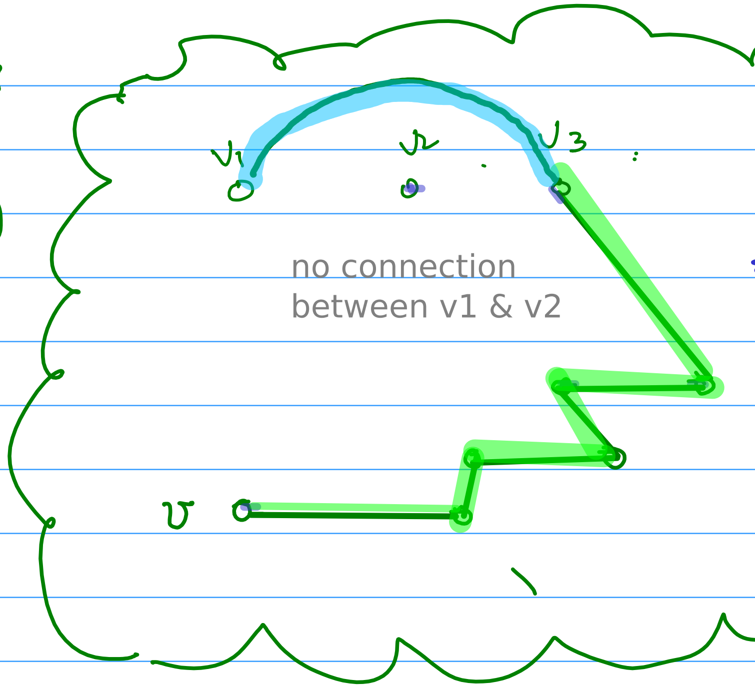
G : Connected ⇒ ($v \rightarrow v_1$ path 존재함) p



$p' =$ a part of p, whose vertex & node are in G'

① $v \rightsquigarrow v_1$

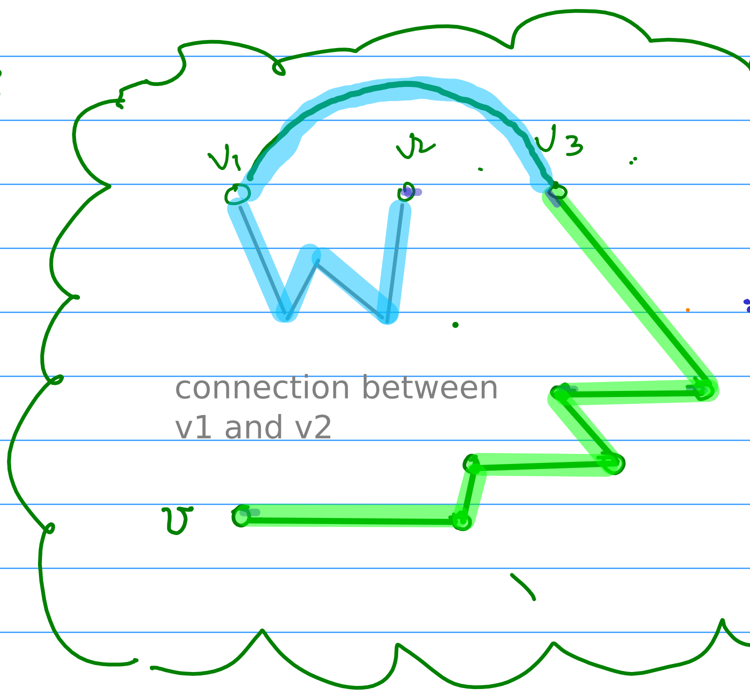
G'



v는 G'의
= v_1과 같은 component에 존재
= v_3와 같은 component에 존재한다

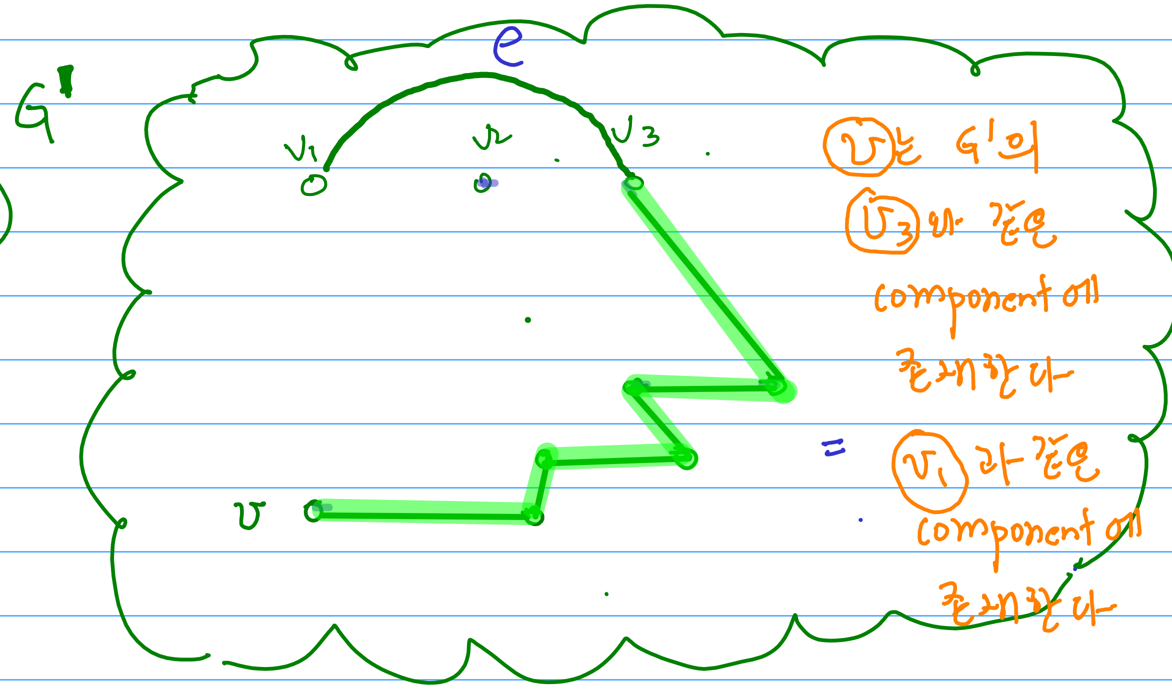
② $v \to v_2$

G'



v는 G'의
= v_2와 같은 component에 존재
v_1, v_3

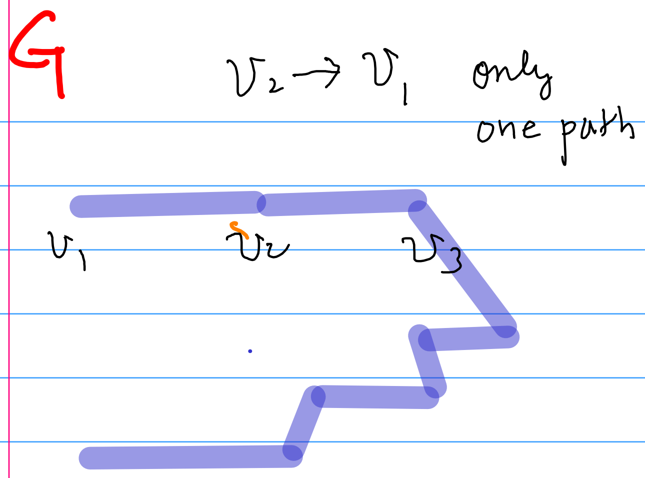
$v \to v_3$

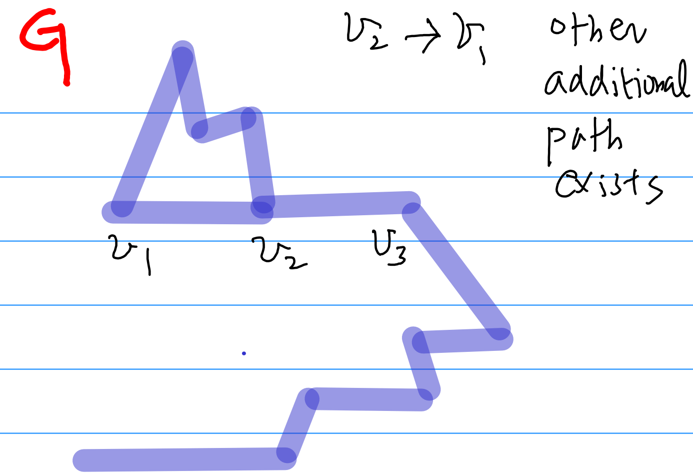


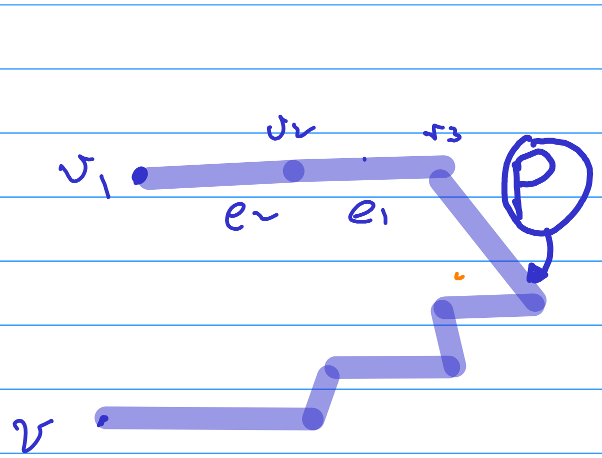
path P : graph G에서 $v \rightsquigarrow v_3 \to v_2 \to v_1$

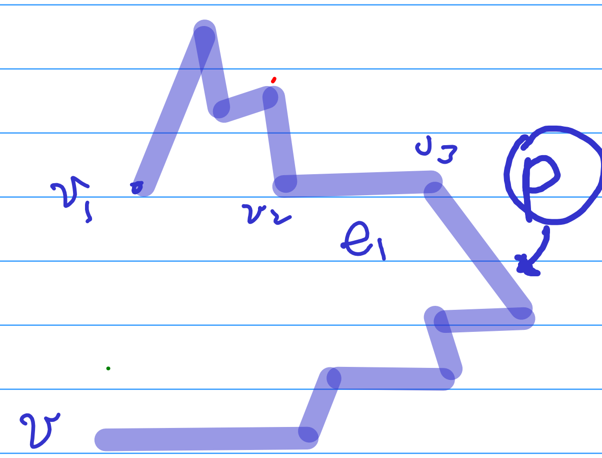
path P' : graph G'에 남아있는 P의 부분 중
v에서 시작하는 path

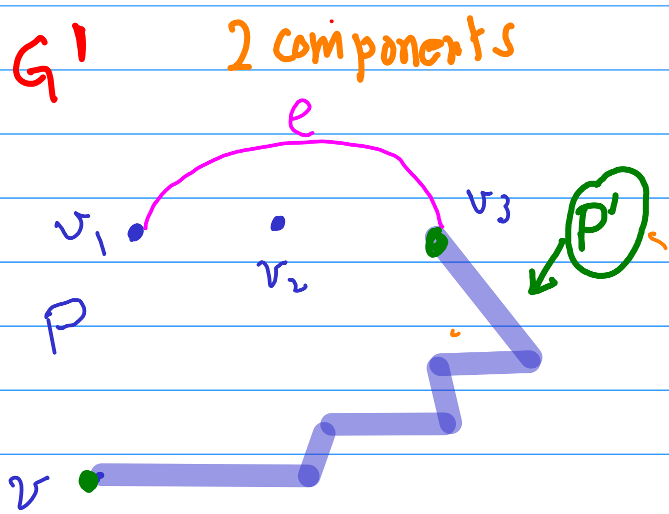
3 cases
$$\begin{cases} v \rightsquigarrow v_3 & \text{P'가 } v_3\text{에서 끝나는 경우} \\ v \rightsquigarrow v_2 & \text{P'가 } v_2\text{에서 끝나는 경우} \\ v \rightsquigarrow v_1 & \text{P'가 } v_1\text{에서 끝나는 경우} \end{cases}$$

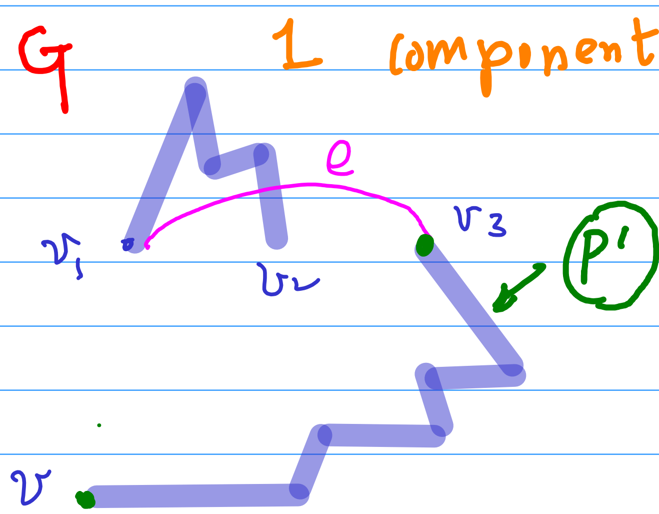






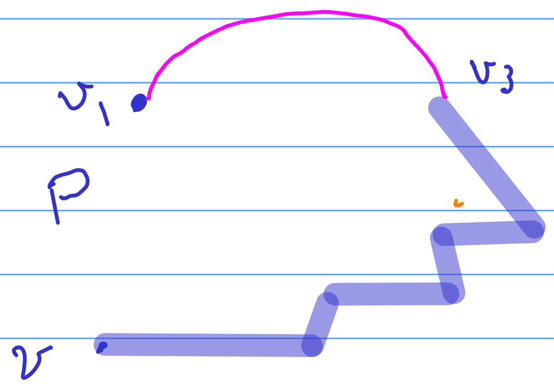


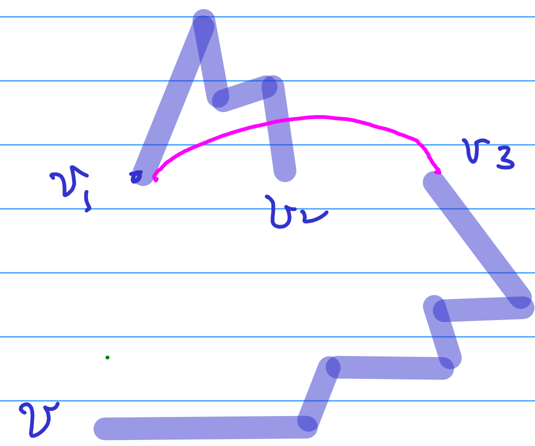




v는 v_1 & v_3 와 같은 component에 있다

v는 v_1, v_2, v_3 와 모두 같은 component에 있다





v, v_1, v_3 만 같은 component에 있다

v, v_1, v_2, v_3 모두 같은 connected comp.

v_2가 연결되지 않음

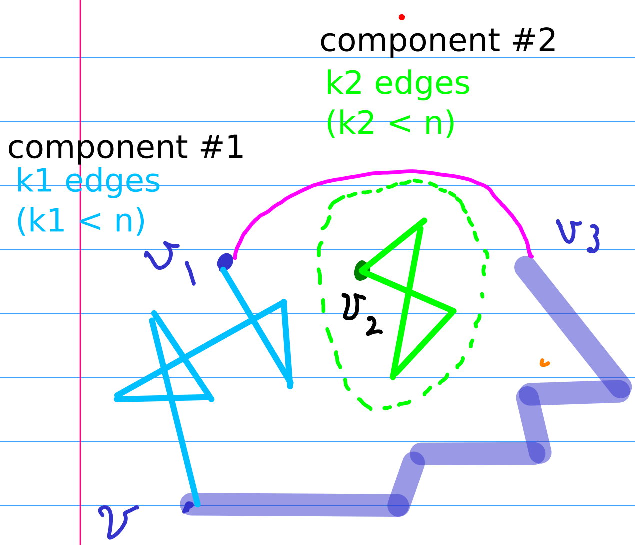
G'에 2개 component 존재

모두 연결됨

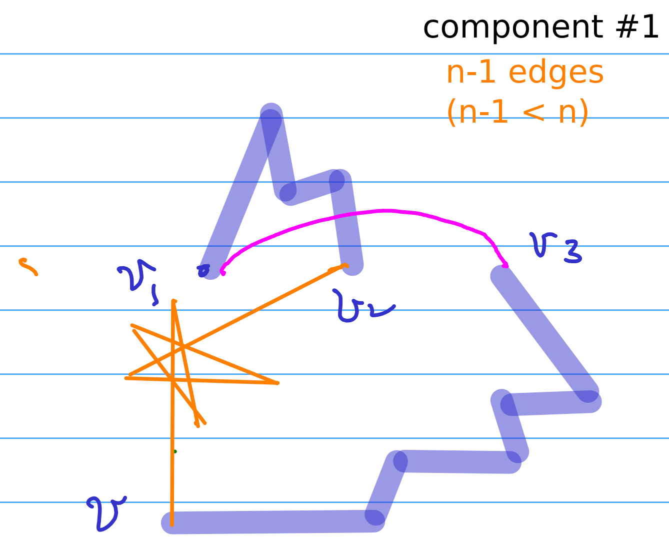
G'에 1개 component

$$\boxed{\text{Connected } G \xRightarrow[\text{add } e]{\text{remove } e_1,\ e_2} G'}$$

G' 2개 component



G' single component



※ G와 G'의 차이점은 e_1 e_2 가 빠지고 e 가 추가된 것

⇒ 모든 component들은 connect 되어있고 even degree이다

n보다 작은 모든 k 에 대하여

$k < n$

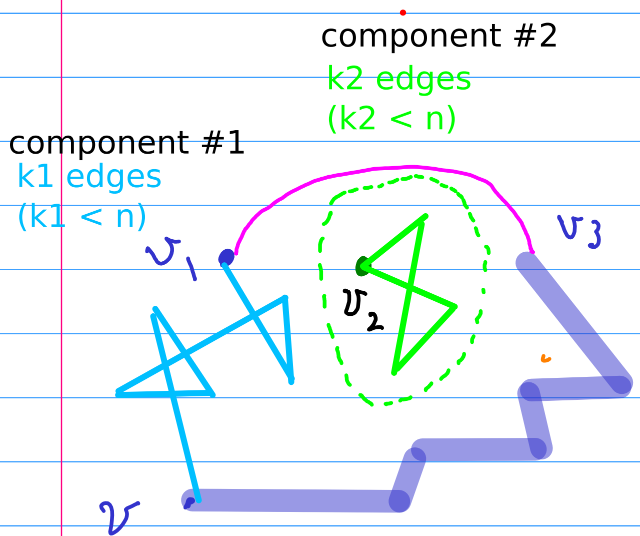
k 개의 edge를 가지고
모든 vertex가 even degree 이고
connected graph ⇒
Euler cycle을 가진다

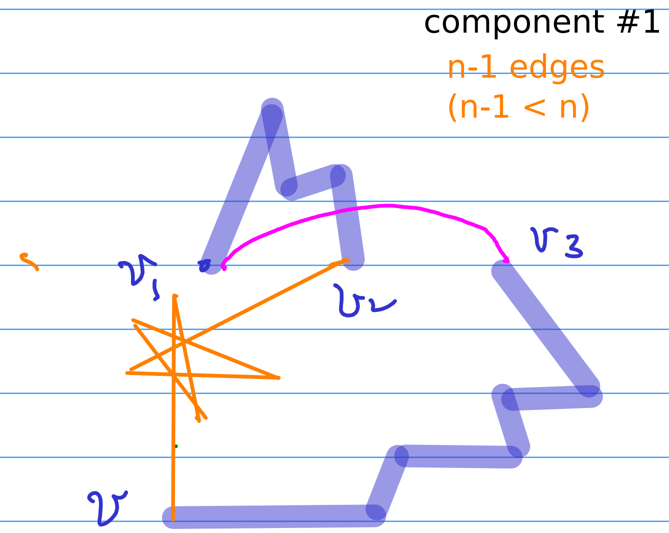
귀납적 가정

n 개의 edge를 가지고
모든 vertex가 even degree 이고
connected graph ⇒
Euler cycle을 가진다

G' 2개 component

G' single component



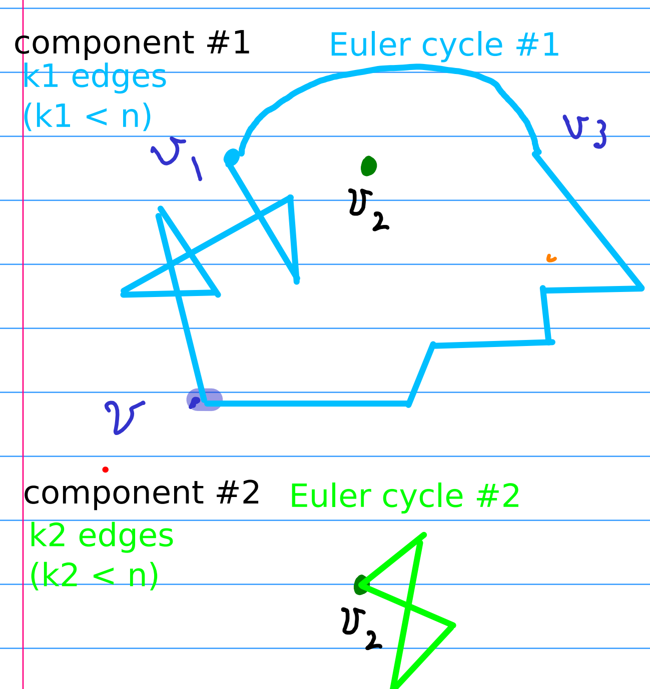


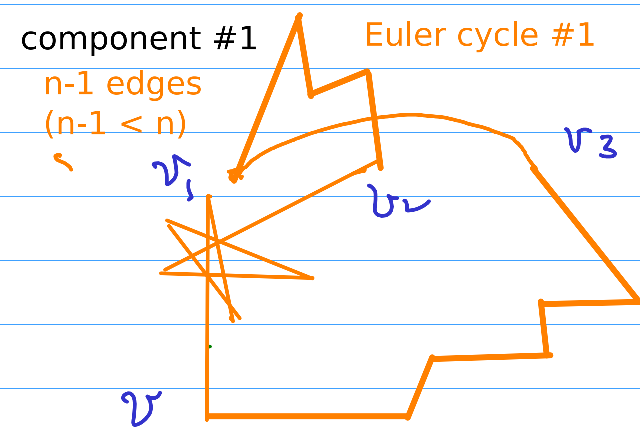
k_1, k_2개 edge
even degree
connected ⇒

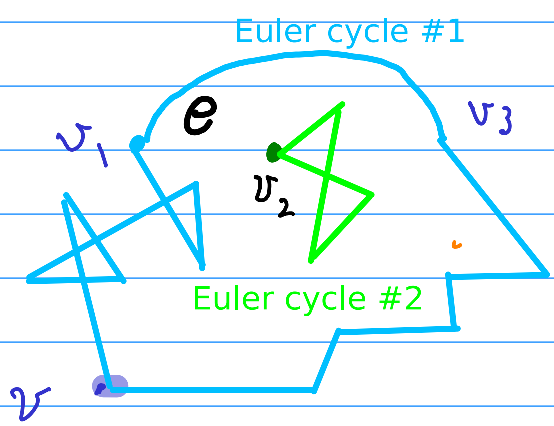
G' 에서 Euler cycle 2개

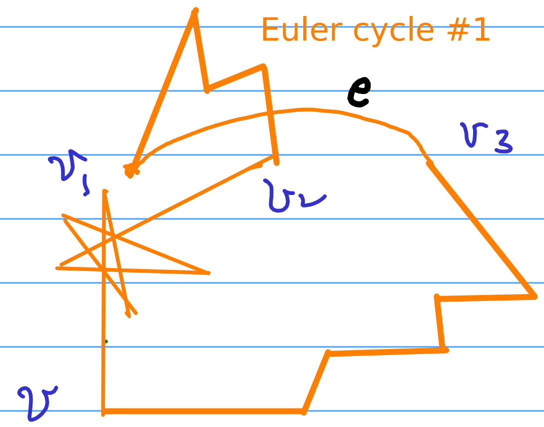
(n-1)개 edge
even degree
connected ⇒

G' 에서 Euler cycle



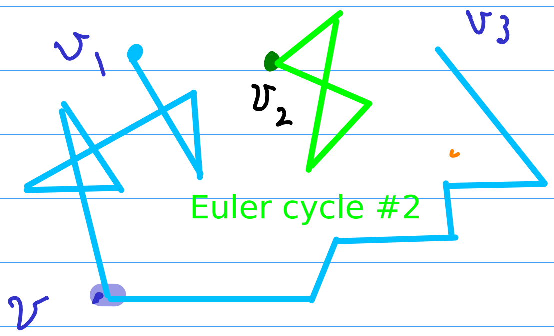


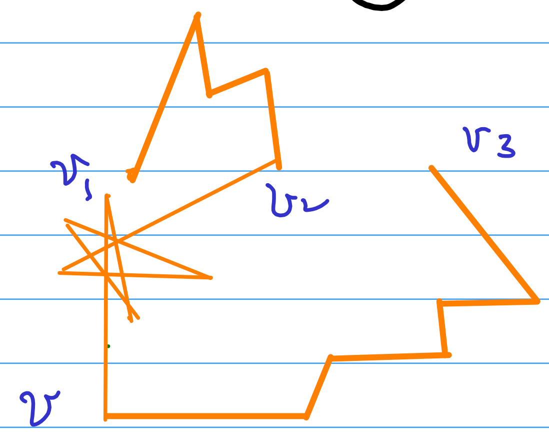




remove e

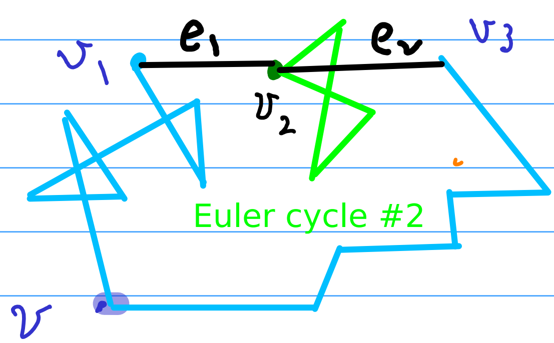
remove e

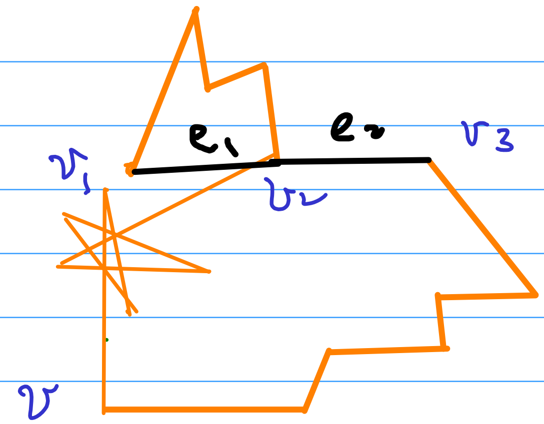




add e_1 e_2

add e_1 e_2

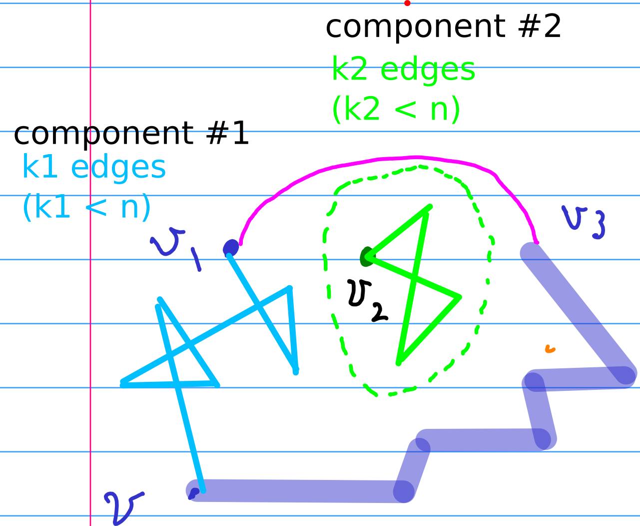




non edge G only Euler cycle

non edge G only Euler cycle

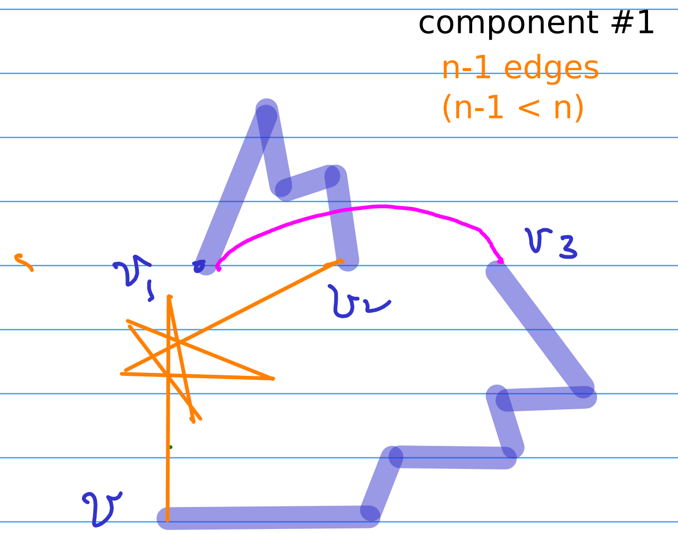
G' 2개 component



k_1, k_2 개 edge
even degree
connected ⇒

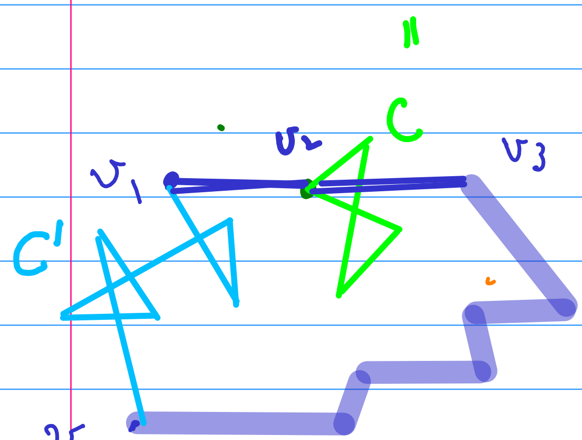
G' 에서 Euler cycle 2개

G' single component



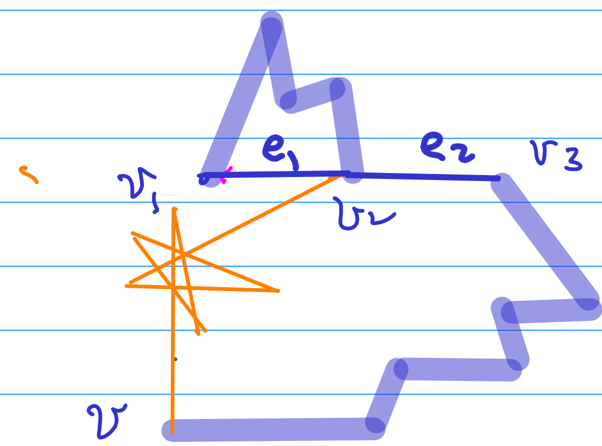
(n-1) 개 edge
even degree
connected ⇒

G' 에서 Euler cycle



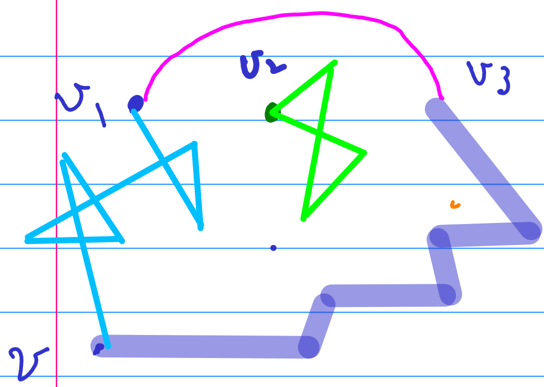
n 개 edge
even degree
connected ⇒

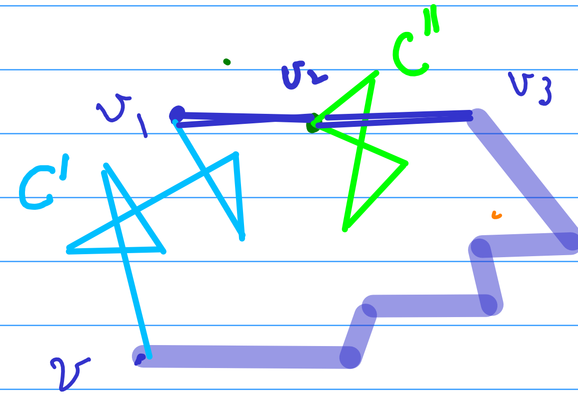
G 에서 Euler cycle



n 개 edge
even degree
connected ⇒

G 에서 Euler cycle





$v_1 \longrightarrow v_3$

$v_3 \longrightarrow v_1$

$\xrightarrow{C'} v_1 \xrightarrow{e_1} v_2 \xrightarrow{C''} v_2 \xrightarrow{e_2} v_3 \xrightarrow{C'}$

$\xrightarrow{C'} v_3 \xrightarrow{e_2} v_2 \xrightarrow{C''} v_2 \xrightarrow{e_1} v_1 \xrightarrow{C'}$

edge 개수 n = 0
n = 1
n = 2
⋮

모든 graph

A connected graph ⇢ only one component
with even degree vertices ⇨ An Euler cycle

꼭지점에 even number of edges

A proof by **induction** on the number of edges in G

A connected graph G with even degree vertices only and *k* edges (*k* < *n*)	⇨	An Euler cycle

A connected graph G with even degree vertices only and *n* edges	⇨	An Euler cycle

모두 connected고 even degree

0 개 edge를 갖는 G. → Euler cycle

1 개 edge를 갖는 G. → Euler cycle

k개 (k<n)

n-2 개 edge를 갖는 G. → Euler cycle

n-1 개 edge를 갖는 G. → Euler cycle

n 개 edge를 갖는 G. → Euler cycle

Base case

trivial case

n = 0 edge

n = 1 edge

euler cycle를 갖는다 ....

n = 2 edge

"n개 edge 갖는 G도 Euler cycle을 가진다"를 보이면 된다

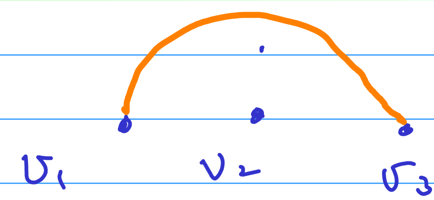
각각 component들은 Euler cycle을 갖는다

→ Euler cycle 갖는다

귀납적 가정 : k개 edge를 갖는 G
- 1개 component
- 꼭지점에 짝수개 edge

이 component들은 edge의 개수 $< n$

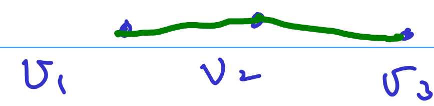
G' : 1개 or 2개 component
꼭지점 마다 짝수개 edge
$(n-1)$개 edge

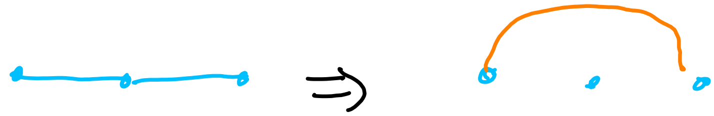


edge 1개

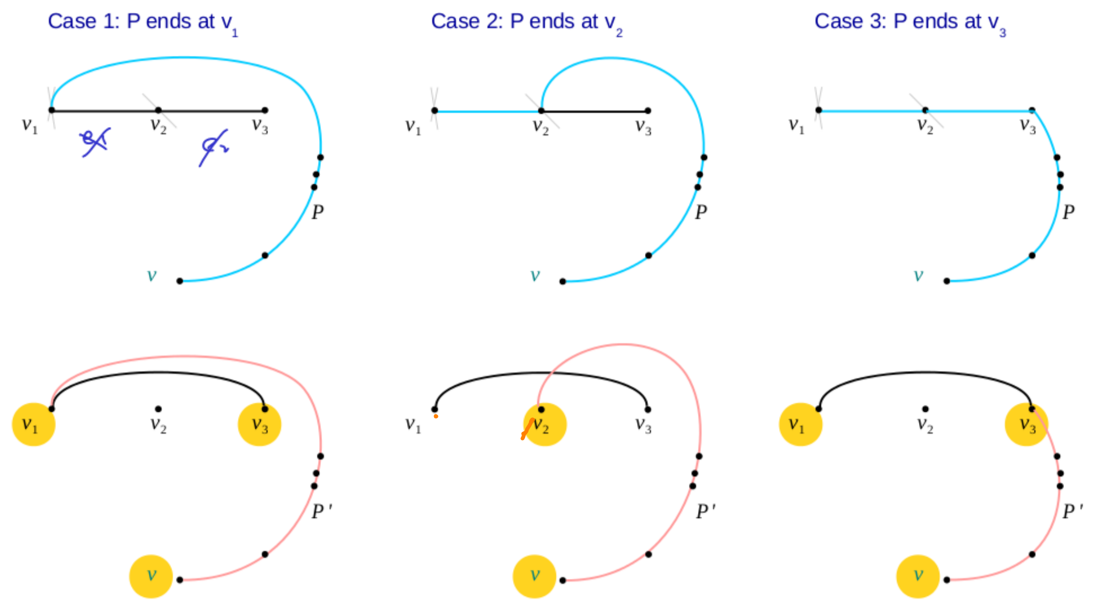
edge 2개

G : 1개 component
꼭지점 마다 짝수개 edge
n개 edge





1개 또는 2개 component가 생긴다.



Johnsonbough, Discrete Mathematics

$\{v, v_1, v_3\} \sim \{v_2\}$

연결되는

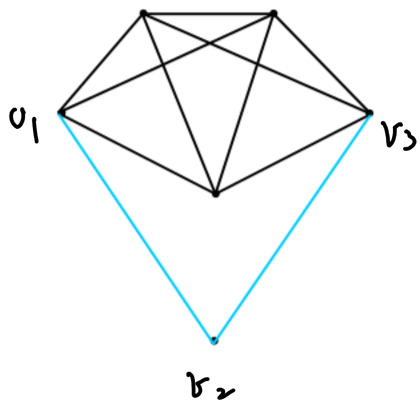
path가 있으면

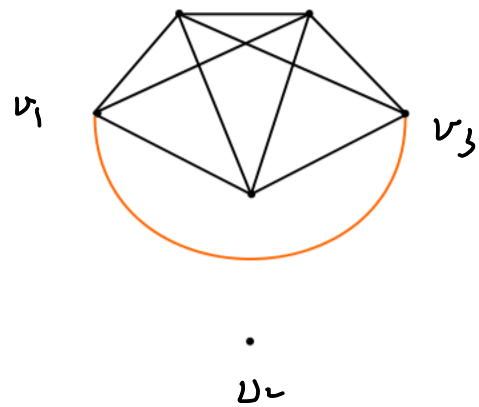
1개 component이고

없으면

2개 component가 된다.

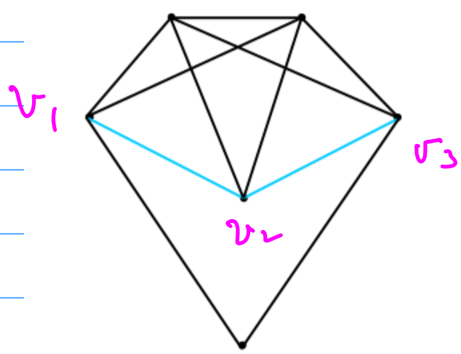
ex)

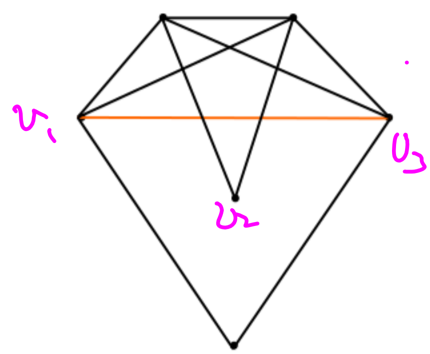




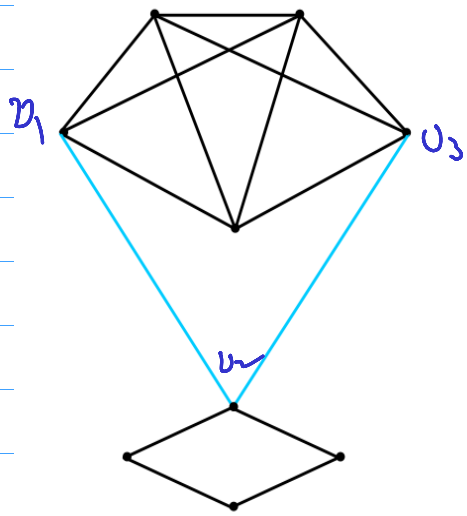
1개 component

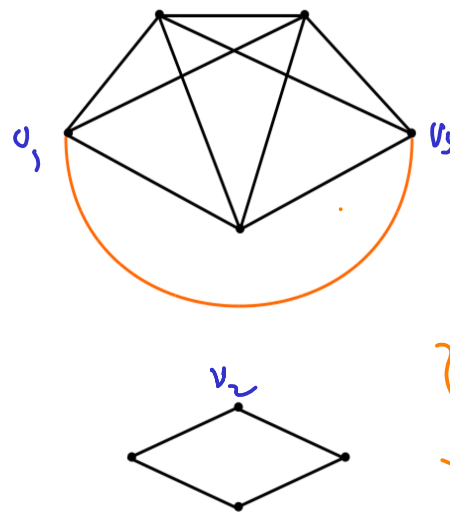
1개 component





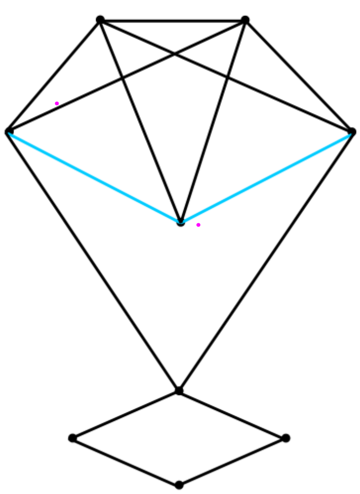
1개 component

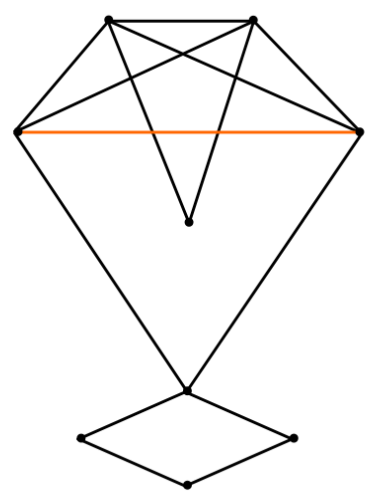




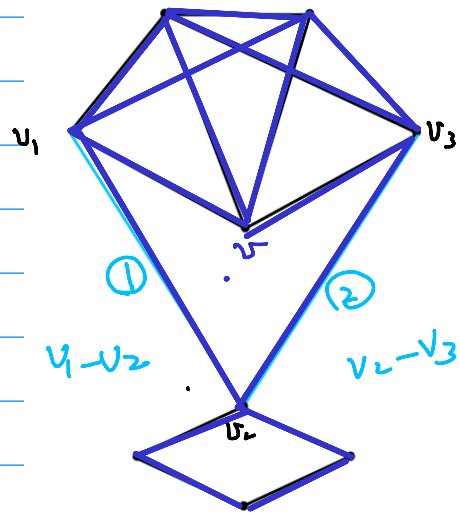
1st component

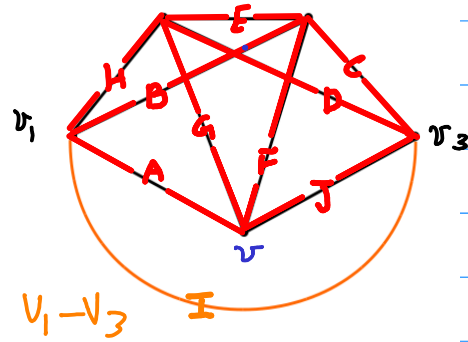
1st component

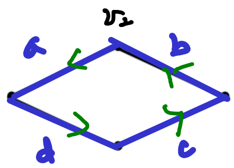




1st component







C'' 은 v_2에서 시작하고 끝나는 Euler cycle

C'

A B C D E F G H I J

10개 짜리 edge G에서 Euler c.

a b c d

C''

4개 짜리 edge G에서 Euler cycle

I

C''

A B C D E F G H 1 a b c d 2 J

15개 짜리 edge G 에서 Euler cycle

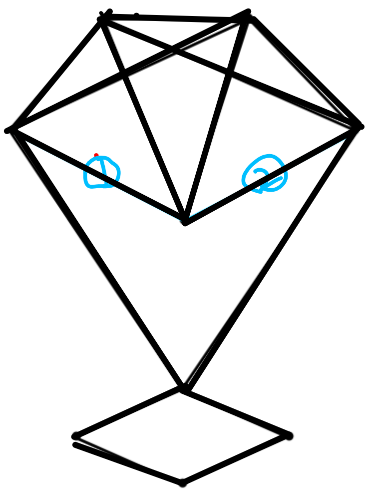
C''

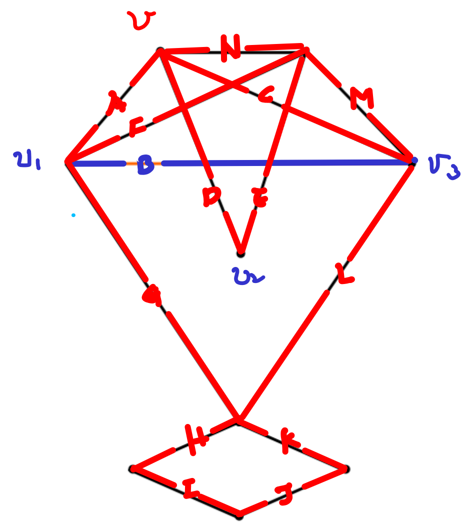
A B C D E F G H 2 a b c d 1 J

15개 짜리 edge G 에서 Euler cycle

A 1 2 C D E F G H I J K L M N

A B C D E F G H I J K L M N





585 p

vertex에 표시하는 문제

$L(v)$가 표시 — temporary mark …… 임시적인 표시

처음에 모든 v는 $L(v) = \infty$

final mark …… 확정적인 표시

length

$L(v)$는 a ⟶ v 까지의 최단 경로.

시작점

temporary

$T = \{v_i\}$ 임시적인 표시를 갖는 vertex들의 집합

확정적인 표시는 그림에서 ○ 원을 사용한다.

$v \notin T$ 인 v에 대하여 $L(v)$는 a ⟶ v 까지의 최단 경로.   시작점

$\boxed{\begin{array}{ll} w & : \text{weight} \\ a, z & : \text{vertex.} \\ L(z) & : \text{the shortest path length from } a \text{ to } z \end{array}}$

dijkstra (w, a, z, L) {

$L(a) = 0$

for every vertex $x \neq a$ $\{L(x) = \infty\}$

$T = \{$ all vertices that are not final $\}$

while $(z \in T)$ { (v)

temporary mark.   Choose $v \in T$ with the smallest $L(v)$

final mark   Eliminate v from T $\boxed{T = T - \{v\}}$

for each neighbor x of v $(x \in T)$

$$\boxed{L(x) = \min\{L(x), L(v) + w(v, x)\}}$$

}

}

Choose $v \in T$ with the smallest $L(v)$

Eliminate v from T $\quad \boxed{T = T - \{v\}}$

for each neighbor x of v $(x \in T)$ $\quad$ Temporary neighbor

$$\boxed{L(x) = \min\{L(x),\ L(v) + w(v,x)\}}$$

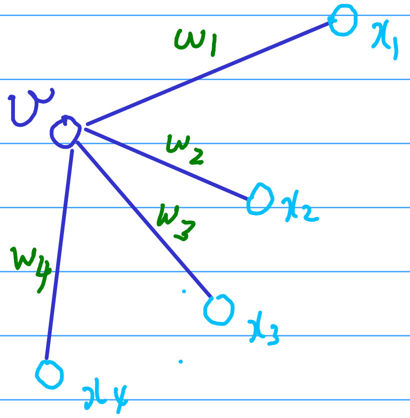
$$T = \{\cdots, v, \cdots\} \qquad v \in T$$

$$= \{v, x_1, \ldots, x_n\} \qquad L(v) \le L(x_i)$$

ε

$$\Downarrow$$

$$T = \{x_1, \ldots, x_n\}$$

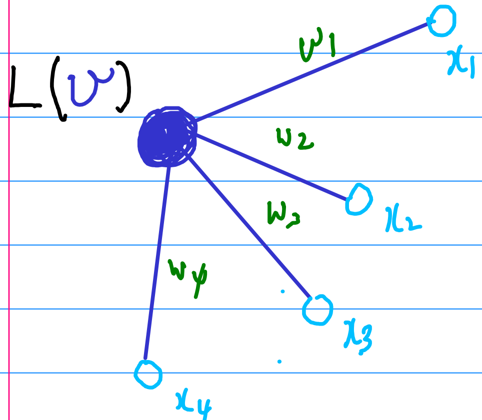


new (updated) $\qquad$ old

$$L(x_1) = \min\{L(v) + w_1,\ L(x_1)\}$$
$$L(x_2) = \min\{L(v) + w_2,\ L(x_2)\}$$
$$L(x_3) = \min\{L(v) + w_3,\ L(x_3)\}$$
$$L(x_4) = \min\{L(v) + w_4,\ L(x_4)\}$$

10 $\qquad$ 12 $\qquad$ 10

12 $\qquad$ 12 $\qquad$ 14

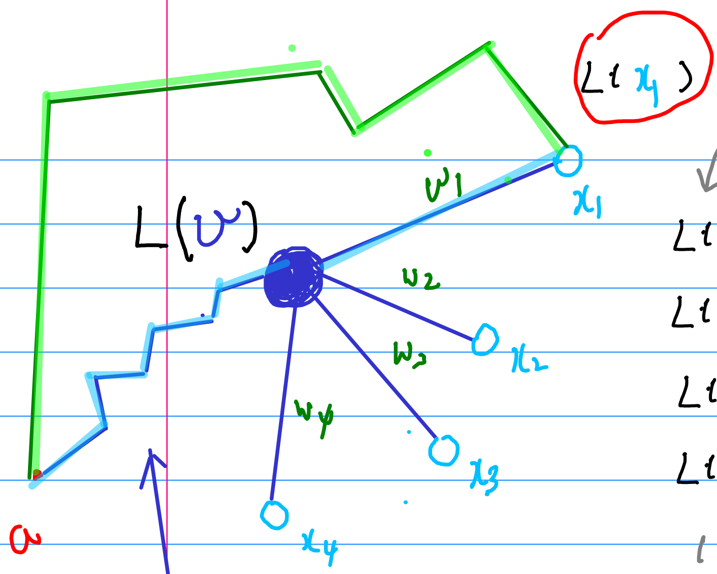


new (updated) $\qquad$ fixed value

$$L(x_1) \le L(v) + w_1$$
$$L(x_2) \le L(v) + w_2$$
$$L(x_3) \le L(v) + w_3$$
$$L(x_4) \le L(v) + w_4$$

$10 \le 12$

$12 \le 12$



a ~ z 최단거리 path
L(v) path length

new (updated)

fixed value

$$
\begin{aligned}
L(x_1) &\le \boxed{L(v) + w_1} \\
L(x_2) &\le L(v) + w_2 \\
L(x_3) &\le L(v) + w_3 \\
L(x_4) &\le L(v) + w_4
\end{aligned}
\qquad \min(\;,\;)
$$

$$
\begin{array}{ccc}
10 & \leq & 12 \\
12 & \leq & 12
\end{array}
$$

L(x1)은 update 되는 path length이다

$$
\begin{array}{ccccc}
L(v) + w_1 & \Leftarrow & L(x_1) & > & L(v) + w_1 \\
L(x_1) & \Leftarrow & L(x_1) & < & L(v) + w_1 \\
L(x_1) & \Leftarrow & L(x_1) & = & L(v) + w_1
\end{array}
$$

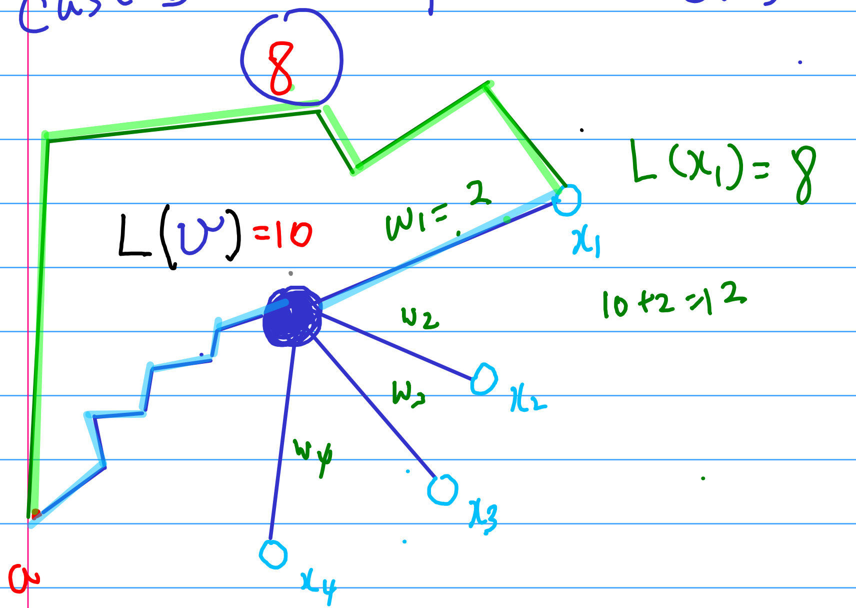
update 되는 값

원래 L(x1) update 하기 전

$a \rightsquigarrow v \to x_1$ path length 값

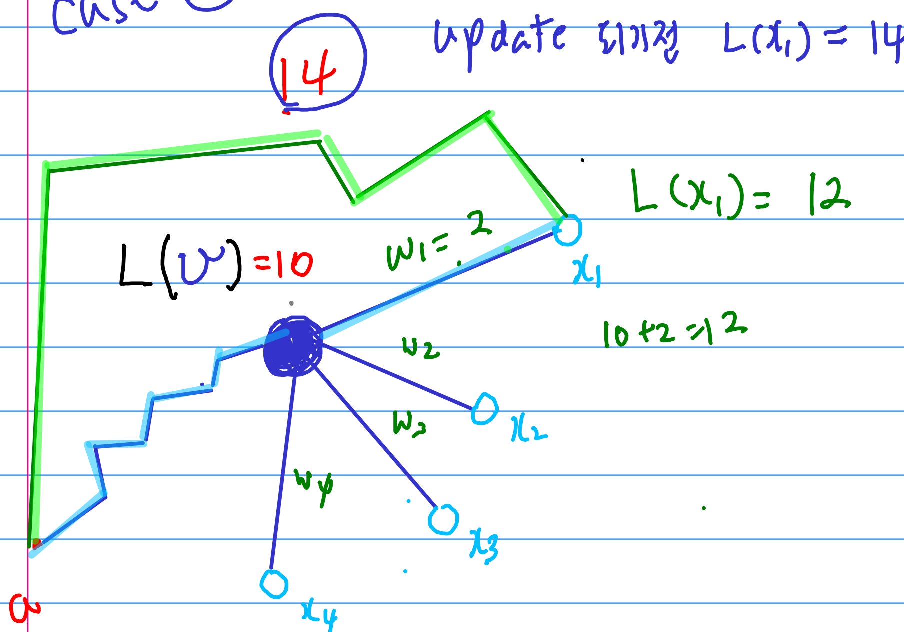
case ①

update 되기전 $L(x_1)=8$



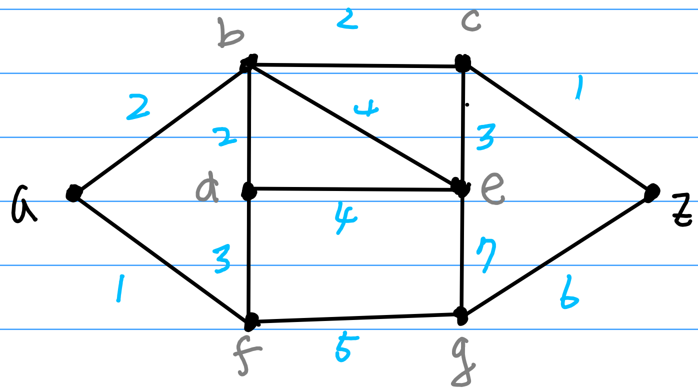
case ②

update 되기전 $L(x_1)=14$



update 된 $L(x_1)$은 $L(v)+w_1$보다 클 수 없다.

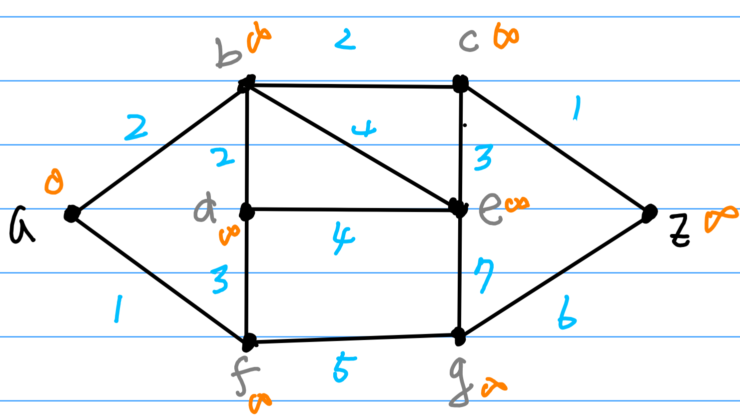
$$L(x_1) \le L(v)+w_1$$



$L(a) = 0$

for every vertex $x \neq a$ $\{L(x) = \infty\}$

$T = \{$ all vertices that are not final $\}$



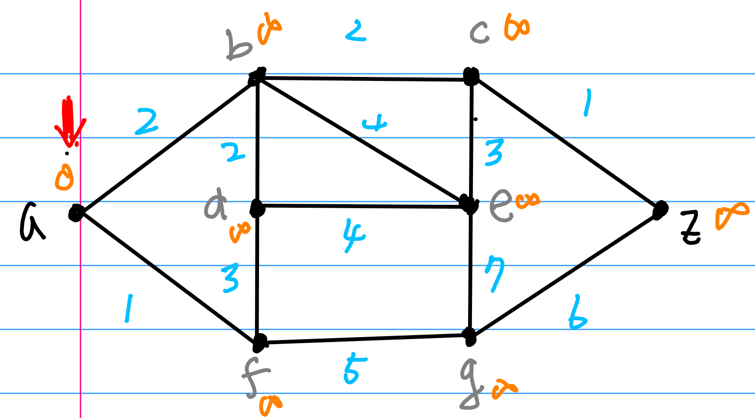
(x) $L(b) = \infty$

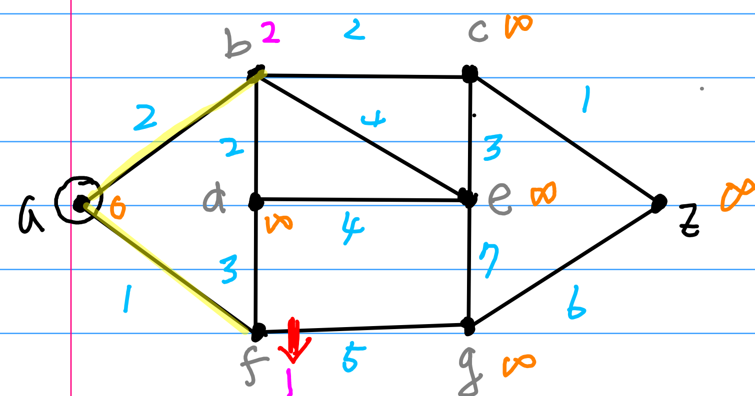
$a \rightsquigarrow b$ ∞

temporary

$T = \{a, b, c, d, e, f, g, z\}$

1



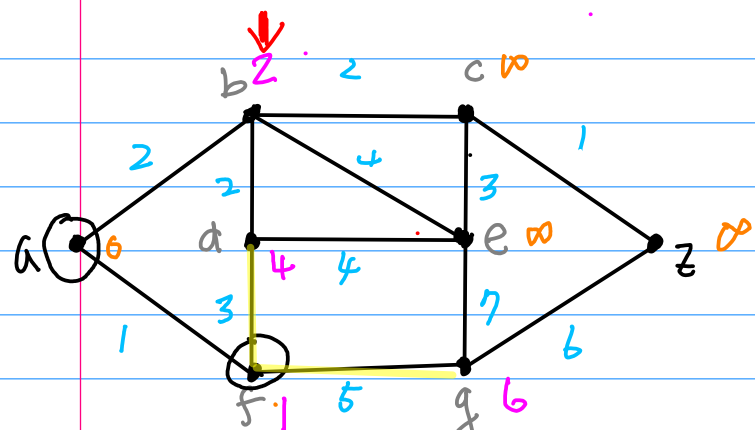


T = {a, b, c, d, e, f, g, z}

minimum L(a) = 0

L(b) = min{∞ , 0+2} = 2

L(f) = min{∞ , 0+1} = 1

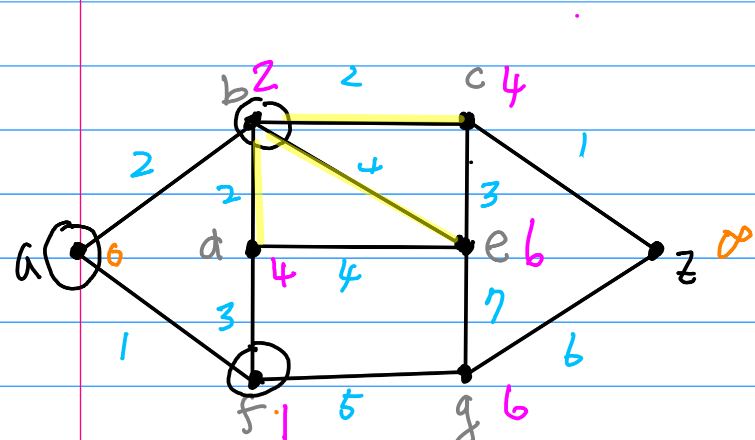


T = {b, c, d, e, f, g, z}

minimum L(f) = 1

L(d) = min{∞ , 1+3} = 4

L(g) = min{∞ , 1+5} = 6



T = {b, c, d, e, g, z}

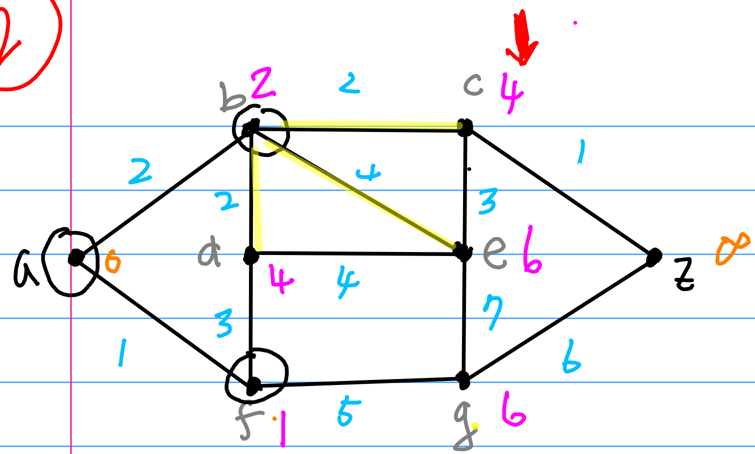
minimum L(b) = 2

L(c) = min{∞ , 2+2} = 4

L(d) = min{4 , 2+2} = 4

L(e) = min{∞ , 2+4} = 6

2



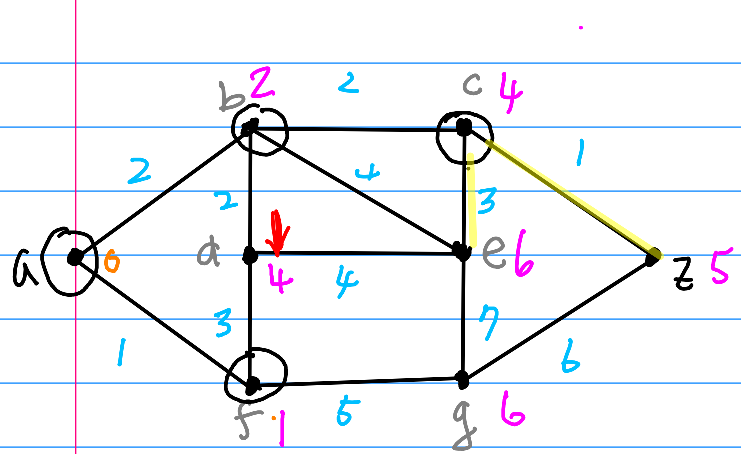
$T = \{b, c, d, e, g, z\}$

minimum $L(b) = 2$

$L(c) = \min\{\infty, 2+2\} = 4$

$L(d) = \min\{4, 2+2\} = 4$

$L(e) = \min\{\infty, 2+4\} = 6$

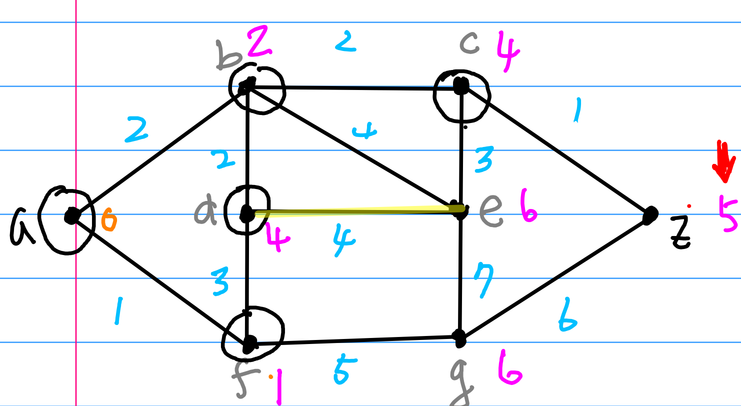


$T = \{c, d, e, g, z\}$

minimum $L(c) = 4$

$L(e) = \min\{6, 4+3\} = 6$

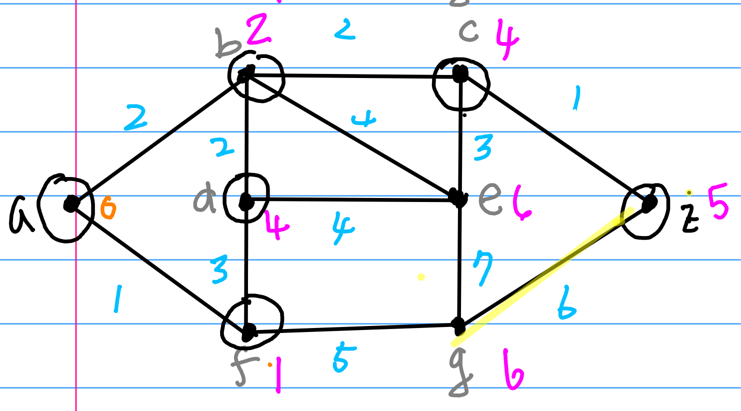
$L(z) = \min\{\infty, 4+1\} = 5$



$T = \{d, e, g, z\}$

minimum $L(d) = 4$

$L(e) = \min\{7, 4+4\} = 7$

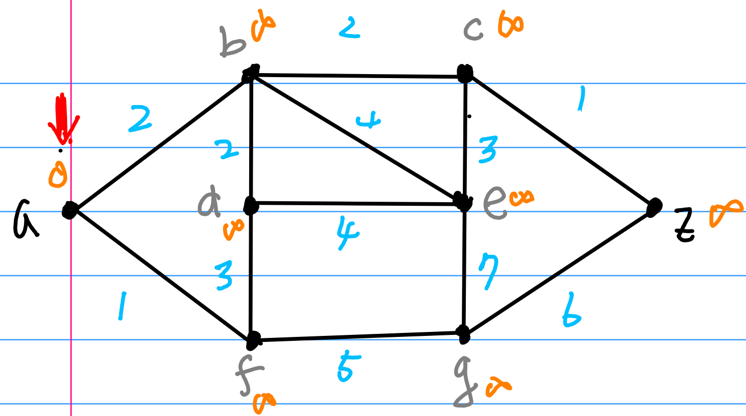


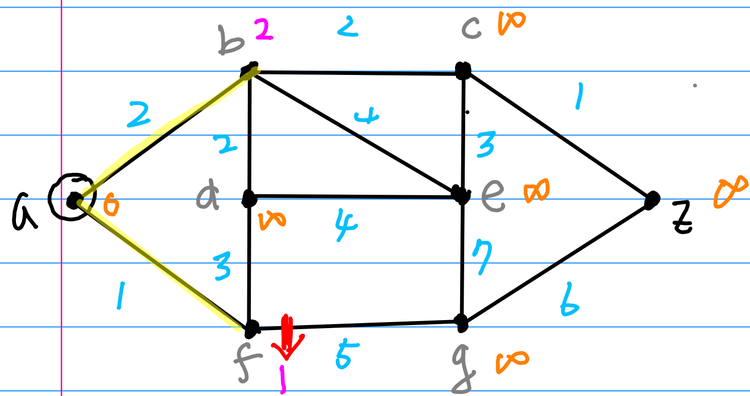
$T = \{e, g, z\}$

minimum $L(z) = 5$

$L(g) = \min\{6, 5+6\} = 6$

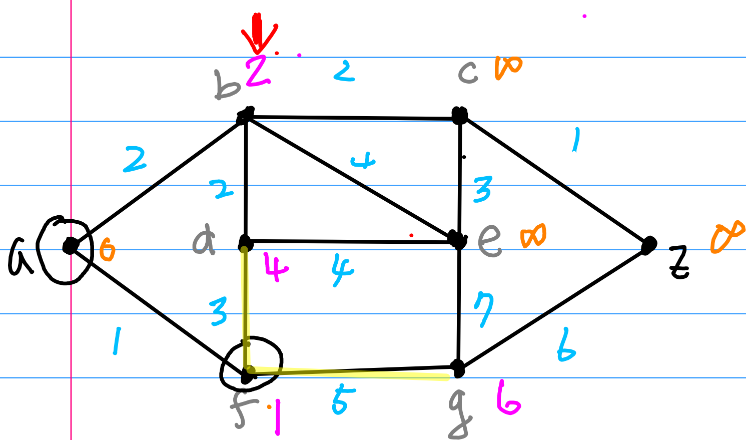
3





$T = \{a, b, c, d, e, f, g, z\}$

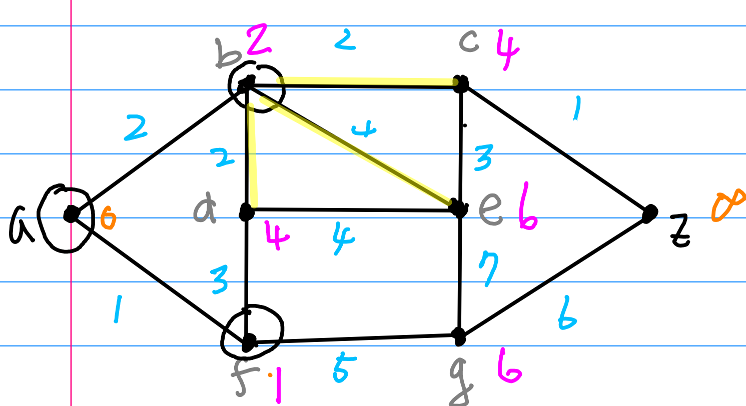
$L(a) = 0$ 1



$T = \{b, c, d, e, f, g, z\}$

$L(a) = 0$ 1

$L(f) = 1$ 2



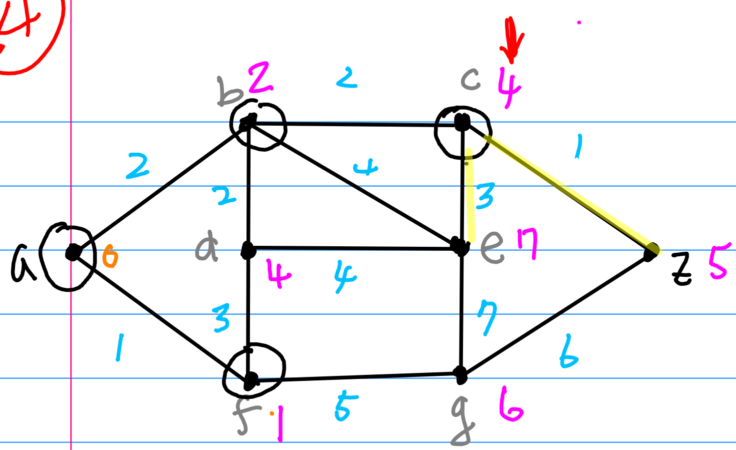
$T = \{b, c, d, e, g, z\}$

$L(a) = 0$ 1

$L(f) = 1$ 2

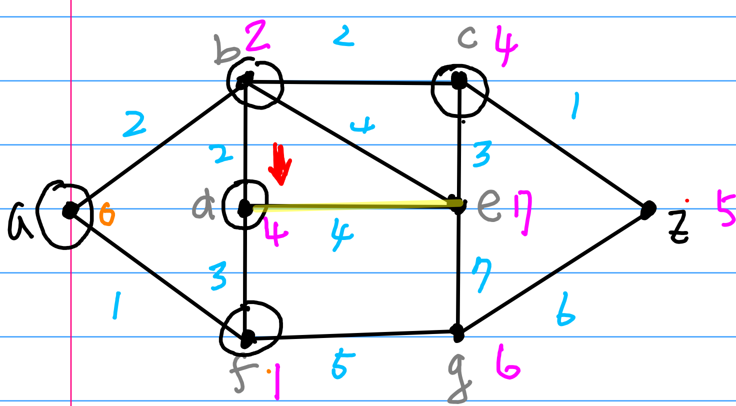
$L(b) = 2$ 3

4



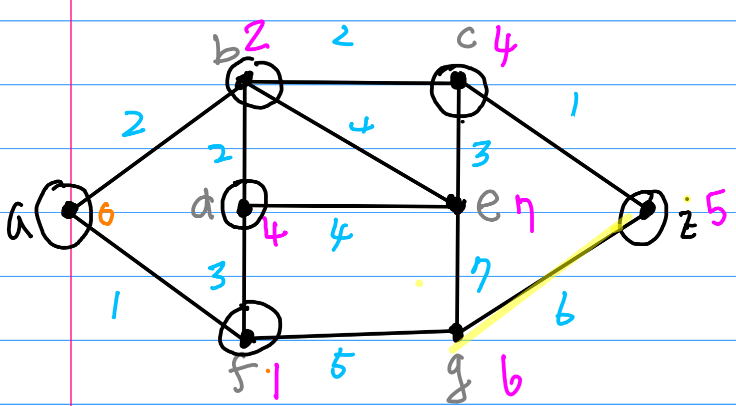
$T = \{ c, d, e, g, z \}$

$L(a) = 0$ ①
$L(f) = 1$ ②
$L(b) = 2$ ③
$L(c) = 4$ ④



$T = \{ d, e, g, z \}$

$L(a) = 0$ ①
$L(f) = 1$ ②
$L(b) = 2$ ③
$L(c) = 4$ ④
$L(d) = 4$ ⑤



$T = \{ e, g, z \}$

$L(a) = 0$ ①
$L(f) = 1$ ②
$L(b) = 2$ ③
$L(c) = 4$ ④
$L(d) = 4$ ⑤
$L(z) = 5$ ⑥

Dijkstra's shortest-path algorithm correctly finds the length of a shortest path from a to z.

mathematical induction on i

prove this ↓

$\boxed{\begin{array}{l} \text{the } i\text{-th time to choose } v \text{ with the minimum } L \\ L(v): \text{ the length of a shortest path } (a, v) \end{array}}$

If the above is true, then when z is chosen,

$L(z)$: the length of a shortest path (a, z)

⇒ the algorithm works!

Basic step $i=1$

initialization : only $L(a)=0$,
$L(v)=\infty \quad (v \neq a)$

Choose $L(a)=0$ ... the length of the shortest path (a, a)

Inductive step i

$k<i$ assume this is true

$$\boxed{\begin{array}{l}\text{the } k\text{-th time to choose } v \text{ with the minimum } L \\ L(v) : \text{the length of a shortest path } (a, v)\end{array}}$$

then, this is also true

$$\boxed{\begin{array}{l}\text{the } i\text{-th time to choose } v \text{ with the minimum } L \\ L(v) : \text{the length of a shortest path } (a, v)\end{array}}$$

the i-th time to choose v with the minimum L

$L(v)$: the length of a shortest path (a, v)

Suppose

the i-th time to choose v with the minimum L

$v \in T$

$L(v)$ the smallest one

Let's show

if there is a path $P(a, w) < L(v)$
then $w \notin T$

Proof by contradiction

$\Rightarrow$ If there were a path from a to v
whose length $P(a, v) < L(v)$
then v would already have been selected
and should have been removed from T

$\Rightarrow$ every path $P(a, v) \geq L(v)$ at least

$\Rightarrow$ there is a path from a to v of length $L(v)$
and this is a shortest path from a to v.

Let's show

$$\boxed{\begin{array}{l}\text{if there is a path } P(a, w) < L(v) \\ \text{then } w \notin T\end{array}}$$

Proof by contradiction

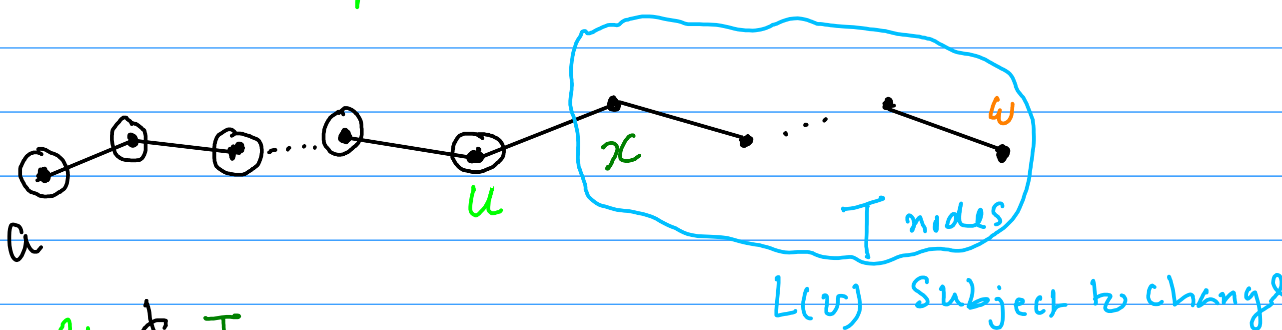
assume this is true

$$\boxed{\begin{array}{l}\text{if there is a path } P(a, w) < L(v) \\ \text{then } w \in T\end{array}}$$

let P : a shortest path from a to w

x : a vertex nearest a on P that is in T

u : a predecessor of x on P



$u \notin T$

therefore u was chosen

during the previous iteration

by inductive assumption

$L(u)$: the length of a shortest path (a, u)

$$\underbrace{L(x) \leq L(u) + w(u, x)} \leq P(a, w) < L(v)$$

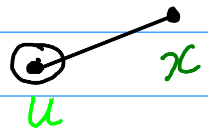
update 결정

min 값 결정

wrong assumption

$\boxed{\text{if there is a path } P(a,w) < L(v) \text{ then } w \in T}$

$$\underbrace{L(x) \leqslant L(u) + w(u,x)}_{\text{update equation}} \leqslant \boxed{P(a,w) < L(v)}$$



~~$v \in T$~~

~~$L(v)$: the smallest~~

$x \in T$

$L(x)$: the smallest

Contradiction

if there is a path from a to a vertex w whose length is less than L(v), then w is not in T

Gray Code

1 bit	2-bit	3-bit	4-bit
0	00	000	0000
1	01	001	0001
	11	011	0011
	10	010	0010
		110	0110
		111	0111
		101	0101
		100	0100
			1100
			1101
			1111
			1110
			1010
			1011
			1001
			1000

binary

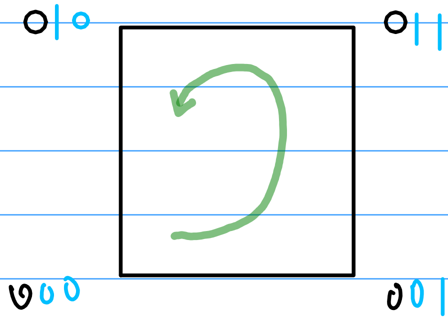
00	000	0000
01	001	0001
10	010	0010
11	011	0011
	100	0100
	101	0101
	110	0110
	111	0111
		1000
		1001
		1010
		1011
		1100
		1101
		1110
		1111

decimal		binary
0	$\longleftrightarrow$	000
1	$\longleftrightarrow$	001
2	$\longleftrightarrow$	010
3	$\longleftrightarrow$	011
4	$\longleftrightarrow$	100
5	$\longleftrightarrow$	101
6	$\longleftrightarrow$	101
7	$\longleftrightarrow$	111

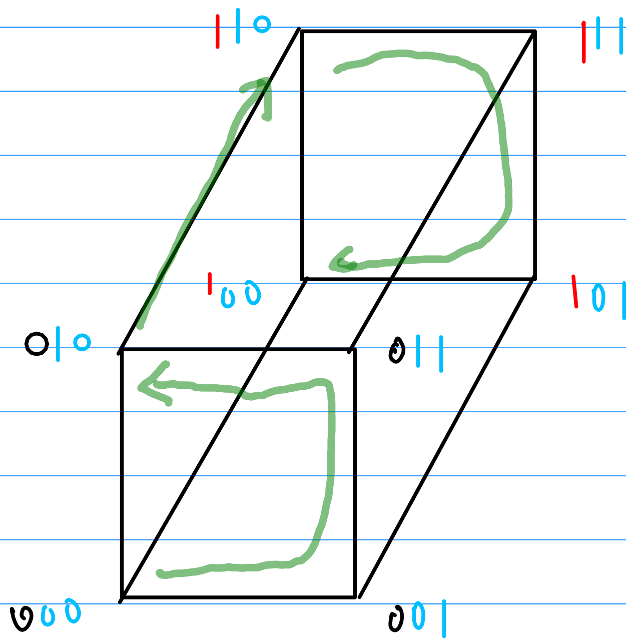
decimal		Gray Code
0	$\longleftrightarrow$	000
1	$\longleftrightarrow$	001
2	$\longleftrightarrow$	011
3	$\longleftrightarrow$	010
4	$\longleftrightarrow$	110
5	$\longleftrightarrow$	100
6	$\longleftrightarrow$	101
7	$\longleftrightarrow$	100

4- cube

2- cube



3-cube



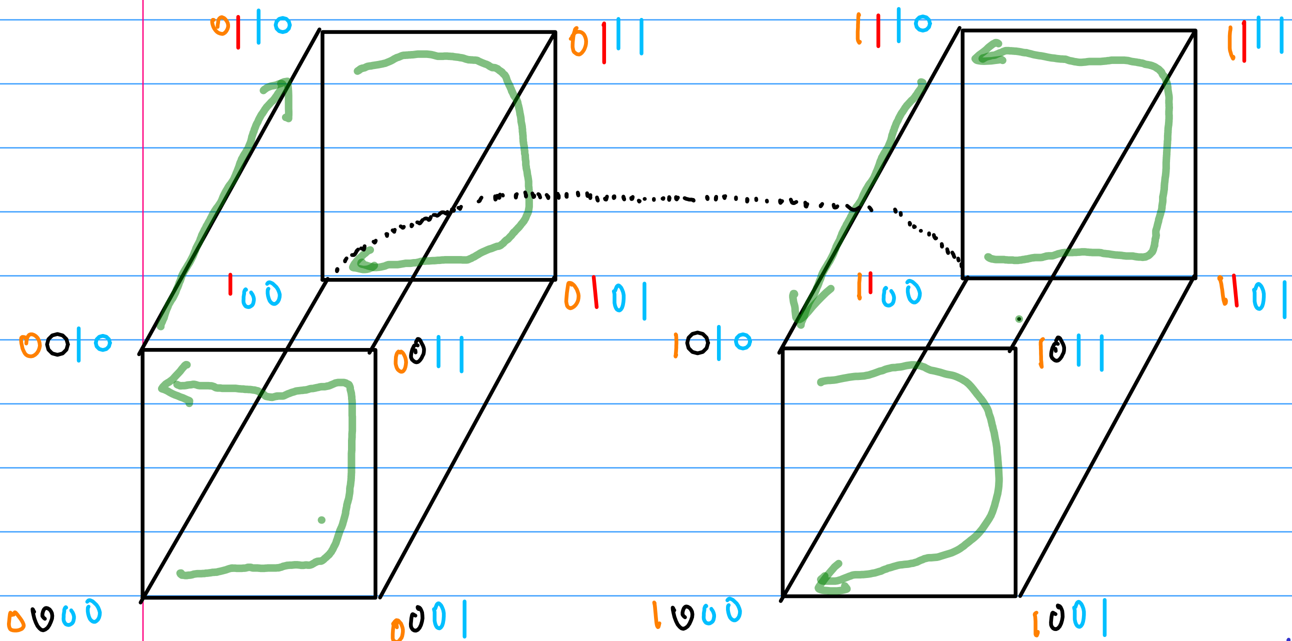
3-bit

0 00
0 01
0 11
0 10

---

1 10
1 11
1 01
1 00

4- cube



1 100
1 101
1 111
1 110
1 010
1 011
1 001

4-bit
0 000
0 001
0 011
0 010
0 110
0 111
0 101
0 100

---

1 100
1 101
1 111
1 110
1 010
1 011
1 001
1 000